Scalable Machine Learning Framework for Behavior-Based Access Control

Jeffrey Cleveland
Raytheon BBN Technologies
Cambridge, MA
jcleveland@bbn.com

Michael Jay Mayhew
US Air Force Research Laboratory
Rome, NY
Michael.mayhew@rl.af.mil

Aaron Adler
Raytheon BBN Technologies
Cambridge, MA
aadler@bbn.com

Michael Atighetchi
Raytheon BBN Technologies
Cambridge, MA
matighet@bbn.com

***Abstract* — Today's activities in cyber space are more connected than ever before, driven by the ability to dynamically interact and share information with a changing set of partners over a wide variety of networks. The success of approaches aimed at securing the infrastructure has changed the threat profile to point where the biggest threat to the US cyber infrastructure is posed by targeted cyber attacks. The Behavior-Based Access Control (BBAC) effort has been investigating means to increase resilience against these attacks. Using statistical machine learning, BBAC (a) analyzes behaviors of insiders pursuing targeted attacks and (b) assesses trustworthiness of information to support real-time decision making about information sharing. The scope of this paper is to describe the challenge of processing disparate cyber security information at scale, together with an architecture and work-in-progress prototype implementation for a cloud framework supporting a strategic combination of stream and batch processing.**

Keywords: security reasoning, machine learning, trust management, access control, cloud computing

I. INTRODUCTION

In current enterprise environments, information is becoming more readily accessible across a wide range of interconnected systems. However, observable behaviors of actors are not explicitly measured after passing initiation authentication, and trustworthiness of documents is only controlled through coarse-grained authorization policies. As a result, mission participants frequently operate unaware of how the latest security events may have impacted the trustworthiness of the information itself and partners participating in exchanges.

While cyber security monitoring systems have significantly evolved over the last decade, these systems still face a number of limitations. First, currently deployed monitoring solutions tend to be signature-based and narrowly focused on specific parts of the overall systems. Examples include the Bro [1] network intrusion detection system and the Host Based Intrusion Detection System (HBSS) [2]. This leaves more sophisticated attacks unhandled, such as 0-day attacks for which signatures are unknown, or insider attacks for which detection requires correlation across systems and layers.

Second, current access control is based on static policies that tie crypto credentials to attributes that are used by access control rules. Dynamic events, such as subversion of credentials (e.g., theft of a Smart Card [3] such as the Common Access Card [4]) or changes in actor behaviors (e.g., insiders performing illegitimate actions within their privilege realm), are not addressed at all, leaving systems vulnerable for a considerable period.

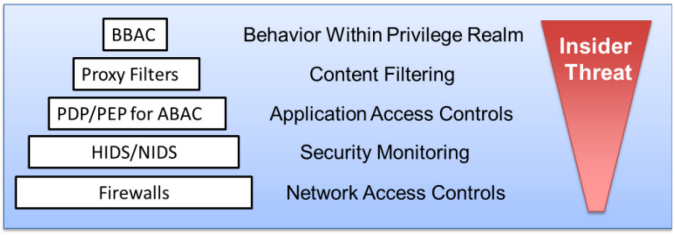

Figure 1. BBAC enables a Layered Defense-in-Depth

BBAC works in conjunction with existing underlying defenses to implement a **defense-in-depth** posture, as shown in Figure 1, and specifically focuses on detecting and mitigating targeted attacks initiated by adversaries with access within privilege realms.

BBAC also enables the concept of **compensating controls**, as visualized in Figure 2, whereby fine-grained controls can avoid the need for coarse-grained deny all policies at lower layers. A lower-layer policy that requires filtering of all traffic to certain networks (e.g., via proxy filters enforcing a static policy as shown in Figure 2a) can be turned into an overall policy whereby the static policy allows interactions if they pass finer-grained checks, e.g., implemented by BBAC (show in Figure 2b).

The sophisticated analysis performed by BBAC at higher layers avoids the need for draconian deny rules at lower layers. By introducing compensating controls, the system becomes more resilient to attacks by restricting access to information only to the services that were involved in the initial compromise vector and successive spread while allowing unaffected systems to continue to participate in information sharing activities and maintain information

Figure 2. BBAC enables Compensating Controls

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
AUG 2013		**00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Scalable Machine Learning Framework for Behavior-Based Access Control	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Raytheon BBN Technologies,10 Moulton Street,Cambridge,MA,02138**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
1st IEEE International Symposium on Resilient Cyber Systems (ISRCS) 2013, August 13-15, 2013, San Francisco, CA.

14. ABSTRACT
Today?s activities in cyber space are more connected than ever before, driven by the ability to dynamically interact and share information with a changing set of partners over a wide variety of networks. The success of approaches aimed at securing the infrastructure has changed the threat profile to point where the biggest threat to the US cyber infrastructure is posed by targeted cyber attacks. The Behavior-Based Access Control (BBAC) effort has been investigating means to increase resilience against these attacks. Using statistical machine learning, BBAC (a) analyzes behaviors of insiders pursuing targeted attacks and (b) assesses trustworthiness of information to support real-time decision making about information sharing. The scope of this paper is to describe the challenge of processing disparate cyber security information at scale, together with an architecture and work-in-progress prototype implementation for a cloud framework supporting a strategic combination of stream and batch processing.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**5**	

To achieve its functionality, BBAC needs to processes vast amounts of data collected by sensors across enterprise environments, including:

- Network flow information, e.g., TCP, and UDP;
- Higher-level transport protocols, e.g., HTTP, XMPP, and SMTP;
- Audit records, e.g., produced by web, OCSP, and DNS servers; and
- Application-level content, e.g., email and chat messages, PDF documents, and wiki pages.

This data is typically collected current enterprise networks but persisted for offline forensics, leading to a situation where "later is too late." BBACs faces the following two main challenges associated with performing real-time processing on the large quantity of data.

Challenge 1: Scalable Training. BBAC uses a strategic combination of supervised and unsupervised machine learning techniques in the form of Support Vector Machines [7] (SVMs), Decision Trees [6], and K-Means clustering [7] (by focusing on these commonly used techniques the scalable processing framework described here has uses beyond the domain of cyber-security). All three of these techniques rely on training data to build up models, and all three techniques have a number of parameters to tweak. This results in a large number of experiments to run to maximize the performance, including varying kernels used for SVMs, parameters to those kernels, and performing n-fold cross validation to determine the resulting accuracy of the various models. In addition, BBAC needs to support dynamic environments in which re-training happens on a daily basis across tens of thousands of observed actors.

Challenge 2: Scalable Classification. BBAC ingests a diverse set of data streams for the purpose of performing real-time behavior classification. This requires a low-latency and highly scalable stream processing framework that can provide elasticity by virtue of load balancing streams over a set of processing nodes. The framework also needs to make it easy to generate higher-level features derived from low-level observables and combine multiple streams with potentially different rates at well-defined junction points.

II. Related Work

There are a number of machine learning frameworks that relate to the BBAC scalable processing framework.

One specific commercial solution is the Pentaho [8] Big Data Analytics Platform. Like BBAC, Pentaho contains the ability to run Weka [9] as part of the overall platform. While Pentaho focuses on interactive data analytics through graphical visualization and reporting, BBAC focuses on the specific purpose of high-throughput classification of cyber security behaviors.

Apache Mahout [10] is an open-source project for scalable machine learning. It provide ready implementations for K-Means clustering following a MapReduce paradigm, but does not provide MapReduce implementations for SVMs, which are the most expensive models to train in BBAC.

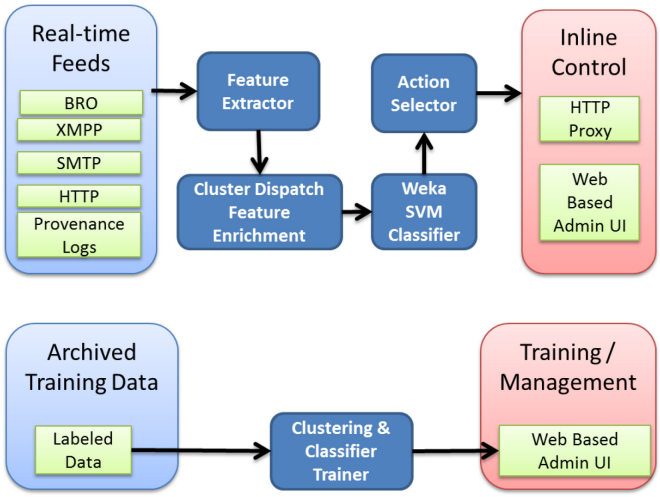

Figure 3. High-level Cloud Architecture

Massive Online Analysis (MOA) [11], developed by the same group as Weka, aims to provide machine learning capabilities to data stream. MOA supports multiple tree learning algorithms, but does not provide support for SVMs for classification.

III. Scalable Machine Learning Framework

Our scalable machine learning framework (illustrated in Figure 3) consists of a distributed batch processing approach to clustering of and training on archived data as well as a distributed stream processing based classifier capable of receiving input from a multitude of data sources. These two views of our system will be elaborated on in Section IIIA and IIIB respectively. In addition to scalability, the framework also provides a large degree of flexibility needed to support drastic changes to the prototype throughout the development cycle and customization between target deployments.

In order to achieve the goals of scalability, flexibility, and reliability, BBAC implements its processing framework using Trident [12], a higher level abstraction on top of the Storm [1] stream processing framework. Storm allows specification of logical topologies of operations on data streams and can transparently distribute stream computation across a large number of compute nodes. Storm also provides reliability through fully fault-tolerant exactly-once messaging semantics, transparently resending information in the case of node failures. Trident adds a higher level of abstraction on top of Storm by providing primitives such as *state*, *query*, *partitions, and micro-batches.*

BBAC strategically combines clustering for segmenting the data into groups that have similar behavior with Support Vector Machines for building a classifier per cluster for the purpose of detecting behavior changes. Clustering serves several purposes in our framework, most importantly faster training times and more accurate and focused classifiers. These advantages all arise because each classifier is trained for a smaller and more similar group of hosts. In addition to boosting accuracy, clustering also allows us to distribute the training and classification to different cluster nodes. BBAC

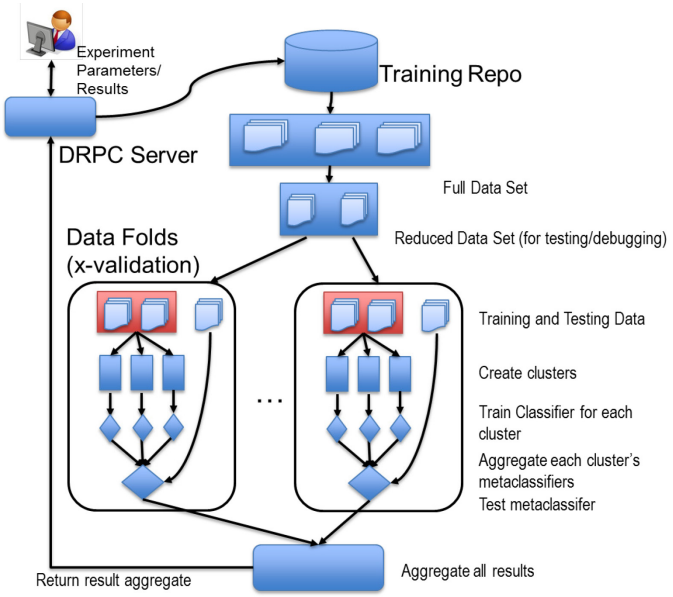

Figure 4. Scalable Training

uses K-Means to form the initial clusters and a prototype implementation of an additional step that uses decision trees.

A. Parallelizing Training Proccesses

Training classifiers within BBAC is a batch process that first involves clustering users/computer behaviors using a large set of data samples, and then training a SVM for each cluster of samples. Figure 4 shows the training architecture.

Training starts with a user sending an experiment request to a Distributed Remote Procedure Call (DRPC) server; the experiment request contains the previously mentioned parameters of the experiment including the location of the desired data set. First, the desired training data is read from a disk or database location and passed to data modifying functions. Currently these functions are limited to reducing the data set (to decrease training time during testing and debugging), but other options such as feature enrichment and data aggregation are planned for future versions of the prototype.

Next, the data is passed to a function that splits the input data into multiple combinations of training and testing data for cross validation. The number of data folds is specified by the user in the experiment request. Note that in order to train a classifier for deployment BBAC skips the cross validation step essentially setting the number of data folds to one. After all data folds are created, the training workflow proceeds to perform clustering on each fold. Next, the workflow trains a classifier for each cluster and then aggregates the resulting classifiers together creating a meta-classifier for each data fold. The corresponding test data is passed to each meta-classifier resulting in a set of statistics describing the performance of each meta-classifier. The meta-classifiers are (optionally) saved to a persistent store, test results are aggregated, and performance metrics are returned to the user.

The current parallelization strategy for the training process is based on distributing the:

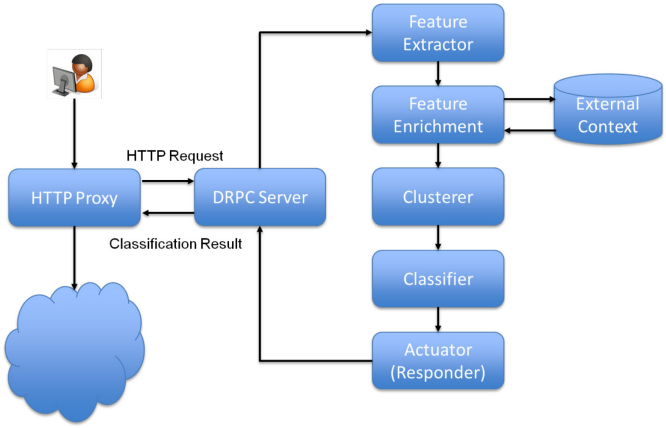

Figure 5. Classification of HTTP Requests

1. building of classifiers per cluster (tens of clusters),
2. changing SVM parameter settings (hundreds of settings), and
3. performing n-fold cross validation experiments for each setting and cluster (with n=10 by default).

Overall, this approach should allow us to train tens of thousands of classifiers in parallel, leading to a significant speedup over sequential training, even with treating the training of a specific classifier instance as an atomic operation.

Initial results from experiments conducted on BBN's Storm cluster seem promising. The recently finished prototype of the processing framework displayed in Figure 4 has moved beyond functional integration issues at this point and can successfully train a set of classifiers on the Storm cluster. However, the first round experiments have not produced the performance gains that were expected, most likely due to unexpected bottlenecks and additional configuration issues that need to be worked out.

B. Parallelizing Classification Proccesses

In order effectively analyze behaviors in DoD enterprise networks, BBAC's classification engine must be able to scale with increasing levels of throughput while also maintaining minimal latency. Figure 5 illustrates a target application where an HTTP proxy passes HTTP requests to BBAC for classification and blocks any request tagged suspicious. For this scenario to be realized BBAC must have latency on the order of tens of milliseconds, a target achieved by the current framework implementation.

The first step in this topology is to parse the input data and extract relevant features. For HTTP requests this involves extracting details such as target URL, time of day, and requester host ID. Other input sources of interest are illustrated in Figure 3 and include Bro data, XMPP traffic, and provenance logs.

The next step takes these features and enriches them using external context. Currently, BBAC issues queries against a cached WhoIS database and identifies the country that corresponds to a provided IP address. Future examples include integration with DNS queries and host subnet information.

The next step involves aggregating samples over a time interval, summarizing certain features per time interval. This step is not currently used for classifying single HTTP requests, as it would likely introduce a level of latency that would make the concept of BBAC working in line with a HTTP proxy infeasible. For other data types, such as TCP connection headers, this aggregation step allows us to analyze how many connections a computer opened over a period of time. Consider the example of a server which normally makes only a few connections to the outside world per hour and is suddenly making thousands of connections per hour. Looking at any one connection request may not reveal suspicious activity, however aggregating connection requests over a given time period may reveal the suspicious behavior. While only hourly summary information is generated for TCP connections currently, it is quite easy in Trident to calculate data summaries over multiple time periods, which time periods are appropriate for the detection of varying attacks is an ongoing area of investigation within the project.

This modular approach to feature extraction, enrichment, and aggregation enables flexibility to experiment with a wide range and types of features during prototype development. Ongoing work includes investigating a wide range of enrichment strategies for more accurate classification.

After the feature enrichment operations are performed, each request/sample is assigned to the correct cluster based on the machine/user ID of the host and passed on to the classifier corresponding to that cluster. Currently these classifiers are Weka-based SVMs. The output of the classification process is passed to at least one of several actuators. The current set of implemented actuators include logging results and updating a system administrator interface; within the context of the HTTP Proxy example this entails returning whether or not the request should be released.

IV. Next Steps

While the authors have developed an initial prototype of the scalable processing framework described in this paper, there are a number of immediate next steps that need to be completed to show the gains in flexibility and scalability that result from it.

Finish Algorithmic Integration: The logic for computing highly summarized features as input to the clustering process needs to be integrated into the Trident framework. Part of this task also involves developing representations of the aggregation logic in a way that allows operators to easily change specifics about the set of features used during aggregation and the time window used.

New Features and Input Streams: Integration of additional features may improve both the clustering and classification results. Examples of new features include WhoIS and DNS information, country information for connection endpoints, and additional information for hosts, such as subnet information. Furthermore, streaming of large volumes of logs into our system will be a focus going forward. The plan is to extend BBAC beyond our current setup of emulating sensor behavior by streaming in static logs collected from such sensors, to developing ingestors that will integrate directly with these network sensors.

Cross Correlating SVMs: Another area of future work is to correlate multiple data sources and classifiers. For example, BBAC currently treats TCP and HTTP logs and analysis separately. The plan is to use Trident's native capability of combining multiple streams for the purpose of combining input steams and/or results in order to correlate activity observed at different levels.

Classifier Resiliency: The parallel classification scheme described by this paper was motivated by the need for scalable processing; it could also provide redundant classification of input data. Such redundancy is often used to enable voting and in doing so would provide resiliency in the face of individual classifiers and/or servers being compromised.

V. Conclusions

Successful defense of cyber infrastructure hinges on the ability to accurately and quickly spot suspicious behaviors across enterprise actors and the information they exchange to identify targeted attacks. BBAC has shown promising accuracy in detecting suspicious activities in TCP connections, HTTP requests, Wikipedia edits, and email exchanges. This paper describes ongoing work focused on scaling BBAC to enterprise environments through a strategic combination of parallelized batch processing for model training and stream processing for classification. Scalability evaluations of the BBAC prototype are currently ongoing and expected to drive further development. The path forward includes finishing algorithmic integration and formulation of new features together with the ability to ingest raw observable from life data stream.

Acknowledgment

The authors would like to acknowledge the support and collaboration of the US Air Force Research Laboratory (AFRL) Information Directorate.

References

[1] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time Computer Networks", 31(23-24), pp. 2435-2463, 1999.
[2] DISA, 2013, Host Based Security System (HBSS), http://www.disa.mil/Services/Information-Assurance/HBS/HBSS
[3] W. Rankl and W. Effing, Smart card handbook. Wiley, 2010.
[4] DoD ID Card Reference Center, 2013, http://www.cac.mil/
[5] Hearst, Marti A., et al. "Support vector machines." Intelligent Systems and their Applications, IEEE 13.4, 18-28,1998.
[6] J. Quinlan, "Induction of Decision Trees", Machine Learning 1: 81-106, Kluwer Academic Publishers, 1986.
[7] J. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297, 1967.
[8] Pantaho Big Data Platform, 2013, http://www.pentaho.com/
[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1, 2009.
[10] Apache Mahout, 2013, http://mahout.apache.org/
[11] MOA, 2013, http://moa.cms.waikato.ac.nz/
[12] Trident, 2013, https://github.com/nathanmarz/storm/wiki/Trident-tutorial
[13] Storm, 2013, http://storm-project.net/